*Ηλίας Σέττας 3150156*

*Χρήστος Γκουρνέλος 3140033*

*Πέτρος Δημητρακόπουλος 3150034*

**Τεχνητή Νοημοσύνη Εργασία 1η: Πρόγραμμα Γυμνασίου**

**1. Βασικές Σημειώσεις και περιορισμοί**

Ο αλγόριθμος που χρησιμοποιείται είναι της μορφής Α\* όπου το σκορ υπολογίζεται με βάση τους σκληρούς περιορισμούς 1 και 2 που δίνονται στην εκφώνηση, τους χαλαρούς 3,4 και 5 αλλά και το βάθος του δένδρου στην συγκεκριμένη κατάσταση.

**Σημείωση 1:** Τα μαθήματα καταχωρούνται όλα μαζί τυχαία στον αρχικό πίνακα schedule χωρίς να παραβιάζονται οι συνολικές ώρες διδασκαλίας τους. Έτσι κατά την παραγωγή παιδιών, όσο αφορά τα μαθήματα, θα χρειαστεί μόνο να κάνουμε αντιμεταθέσεις μεταξύ τους στον πίνακα.

**Σημείωση 2:** Κάθε μάθημα ενός τμήματος (Α1,Α2,…Γ3) πρέπει να διδάσκεται από μόνο 1 καθηγητή καθ’ όλη τη διάρκεια του προγράμματος. Αν 2 ή περισσότεροι καθηγητές διδάσκουν παρόμοια μαθήματα τότε αυτά θα πρέπει να διαχωριστούν μεταξύ τους στα διαφορετικά τμήματα.

**Σημείωση 3:** Σύμφωνα με την προηγούμενη σημείωση θα πρέπει να δίνονται και κατάλληλες ώρες διδασκαλίας στους καθηγητές. Για παράδειγμα έστω ότι 2 καθηγητές διδάσκουν μόνο μαθηματικά Α1 και Α2 και κάθε ένα από αυτά απαιτεί 4 ώρες. Αν ο πρώτος καθηγητής έχει 3 ώρες διαθέσιμες και ο άλλος 5 τότε φυσικά η δημιουργία προγράμματος είναι αδύνατη όπως και στην πραγματικότητα.

**Σημείωση 4:** Στο τέλος το πρόγραμμα παράγει 2 schedule αρχεία. Το ένα είναι txt που περιέχει συνδυασμό κωδικός μαθήματος – κωδικός καθηγητή και το άλλο είναι csv που περιέχει συνδυασμό όνομα μαθήματος – όνομα καθηγητή.

**Σημείωση 5:** Τα αρχεία μαθημάτων που δημιουργήσαμε και χρησιμοποιούμε, είναι από πραγματικό πρόγραμμα σχολείου, έτσι κάθε τάξη θα έχει 32 διδακτικές ώρες και 3 ώρες κενού, πράγμα που σημαίνει ότι δεν θα υπάρχουν παραπάνω από 3 ώρες κενού σε κάθε εβδομάδα.

**Δυνατοί περιορισμοί:**

* Δεν πρέπει να υπάρχουν κενά ανάμεσα σε ώρες διδασκαλίας.
* Κάθε καθηγητής δεν πρέπει να δουλεύει συνεχόμενα πάνω από 2 ώρες.
* Κάθε καθηγητής δεν πρέπει διδάσκει την ίδια στιγμή(ώρα) σε 2 η παραπάνω διαφορετικά τμήματα.
* Οι ώρες διδασκαλίας των καθηγητών δεν πρέπει να ξεπερνάνε τις διαθέσιμες ώρες τους.
* Κάθε μάθημα σε κάθε τμήμα πρέπει να διδάσκεται από μόνο 1 καθηγητή.

Στους παραπάνω περιορισμούς μπορεί να προσέξετε ότι κανείς δεν έχει σχέση με μαθήματα. Αυτό συμβαίνει επειδή στην αρχή κατά την τυχαία κατανομή μαθημάτων στο πρόγραμμα φροντίζουμε να μην υπάρξουν προβλήματα. Ο συνδυασμός του 2ου περιορισμού και του 5ου σημαίνει ότι κάθε μάθημα θα διδάσκεται συνεχόμενα το πολύ 2 ώρες εφόσον θα το διδάσκει ο ίδιος καθηγητής και δεν μπορεί να διδάσκει συνεχόμενα για πάνω από 2 ώρες.

**Χαλαροί περιορισμοί:**

* Οι ώρες διδασκαλίας ενός μαθήματος πρέπει να είναι σχετικά δίκαια κατανεμημένες στο πρόγραμμα.
* Οι ώρες διδασκαλίας για τους καθηγητές κατά τη διάρκεια της εβδομάδας πρέπει να είναι όσο το δυνατόν ομοιόμορφες για όλους τους καθηγητές.
* Οι ώρες διδασκαλίας κάθε μέρας πρέπει να είναι σχετικά ομοιόμορφες κατά τη διάρκεια της εβδομάδας.

**2.1 Ανάγνωση αρχείων**

Η ανάγνωση των αρχείων lessons και teachers γίνεται στην main με τη βοήθεια scanner. Για κάθε αρχείο σε μια επανάληψη διαβάζουμε κάθε γραμμή του αρχείου και καταχωρούμε τα στοιχεία σε 1 κατασκευαστή (Lesson ή Teacher) και στη συνέχεια τον βάζουμε στον πίνακα που διαθέτουμε (lessons ή teachers). Πάντα και στα 2 αρχεία η πρώτη γραμμή αναφέρει τον αριθμό καθηγητών/μαθημάτων ώστε να μπορεί να γίνει από την αρχή η αρχικοποίηση του σχετικού πίνακα και η δέσμευση μνήμης. Όταν διαβαστεί μια γραμμή την σπάμε με split ώστε να πάρουμε σε string την κάθε λέξη της. Στο lessons πρώτα έχουμε τον κωδικό του, μετά το όνομά του, το τμήμα διδασκαλίας του και τέλος της ώρες διδασκαλίας. Στο teachers πρώτα έχουμε τον κωδικό, μετά το όνομα, στη συνέχεια ανάμεσα στο όνομα και την τελευταία λέξη έχουμε τα μαθήματα που μπορεί να διδάξει και τέλος έχουμε τις διαθέσιμες ώρες διδασκαλίας του καθηγητή.

**2.2 Έγγραφη Αρχείων**

Οι εγγραφές γίνονται και αυτές στην main μέσα στις συναρτήσεις WriteTXT και WriteCSV. Το πρόγραμμα θα παραγάγει 1 αρχείο txt στο οποίο για κάθε τμήμα θα παρατίθενται ο κωδικός μαθήματος δίπλα στον κωδικό καθηγητή που το διδάσκει εκείνη την ώρα σε ένα πίνακα 5x7 επί 9 πίνακες (για κάθε τμήμα). Αυτό γίνεται με τη χρήση buffered writer. Για καλύτερη όψη του προγράμματος γίνεται η δημιουργία του csv με τη χρήση file writer όπου σε κάθε κελί υπάρχει το όνομα μαθήματος δίπλα στον καθηγητή που το διδάσκει. Οι στήλες(5) χρησιμοποιούνται για την σημείωση των ημερών ενώ οι γραμμές(7) για τις ώρες.

**3.1 Κατασκευαστής Teacher**

Το αρχείο Teacher.java αποτελεί τον κατασκευαστή της κλάσης καθηγητών. Χρησιμοποιείται ο κωδικός(integer), το όνομα(string), οι ώρες(integer) και ένα array list για την αποθήκευση των κωδικών των μαθημάτων που μπορεί να διδάξει. Κατά την αρχικοποίηση ανατίθενται τιμές σε αυτές τις μεταβλητές ενώ οι αλλαγές και η επιστροφή των τιμών τους γίνεται μέσω συναρτήσεων get και set. Επίσης υπάρχει και η συνάρτηση teaches για έλεγχο του αν υπάρχει ένα συγκεκριμένο στη λίστα του καθηγητή.

**3.2 Κατασκευαστής Lesson**

Το αρχείο Lesson.java αποτελεί τον κατασκευαστή της κλάσης μαθημάτων. Χρησιμοποιείται ο κωδικός(integer), το όνομα(string), οι ώρες(integer), το τμήμα(character) και ένα array list για την αποθήκευση των καθηγητών που μπορούν να διδάξουν αυτό το μάθημα. Κατά την αρχικοποίηση ανατίθενται τιμές σε αυτές τις μεταβλητές ενώ οι αλλαγές και η επιστροφή των τιμών τους γίνεται μέσω συναρτήσεων get και set.

**3.3 Κατασκευαστής Compo**

Η συγκεκριμένη κλάση χρησιμοποιείται στο schedule για κάθε θέση του. Αποθηκεύει τον κωδικό του καθηγητή και του μαθήματος δημιουργώντας έτσι έναν συνδυασμό. Κατά την αρχικοποίηση δίνονται τιμές στους 2 ακεραίους ενώ η επιστροφή των τιμών και η ανάθεση τους μπορεί να γίνει μέσω συναρτήσεων get και set. Με τη συνάρτηση same ελέγχουμε αν ο συγκεκριμένος συνδυασμός είναι ίδιος με κάποιον άλλο ελέγχοντας τις τιμές των 2 κωδικών τους.

**4.1 Program**

Στην αρχή γίνεται αρχικοποίηση του πίνακα schedule(5x7x9) και ανάθεση μηδενικών τιμών σε κάθε θέση του. Σημειώνεται ότι μηδενικός κωδικός μαθήματος και καθηγητή ισούται με κενό στην συγκεκριμένη ώρα του προγράμματος. Μετά γίνεται η ανάγνωση των αρχείων όπως αναφέρθηκε και πιο πάνω και αφού τυπωθούν τα δεδομένα μπαίνει σε μια επανάληψη το πολύ 10 φορές η αρχικοποιεί μια κατάσταση State χρησιμοποιώντας την συνάρτηση Randomize. Εφόσον γίνει η αρχικοποίηση, ορίζουμε το βάθος του State που δημιουργήσαμε ίσο με το μηδέν (διότι είναι το 1ο παιδί του «δέντρου» που δημιουργούμε) και έπειτα υπολογίζουμε το Score του με την ευρετική, καθώς και το F του. Όταν τελειώσουν όλα αυτά, θα δημιουργηθεί ένα αντικείμενο A\_Star και θα ξεκινήσει η αναζήτηση για ένα πρόγραμμα το οποίο δεν παραβιάζει κανέναν δυνατό περιορισμό μέσω της συνάρτησης Search. Αν μας επιστραφεί τελική κατάσταση με σκορ ίσο με -100 τότε δεν έχει βρεθεί λύση και γίνεται επανάληψη της διαδικασίας. Σημειώνεται ότι το -100 δίνεται ως τελική κατάσταση εάν ο αριθμός παιδιών είναι πολύ μεγάλος, έτσι ώστε το πρόγραμμα να μπορεί να τρέχει και με 4GB RAM. Επίσης σε περιπτώσεις που το μέγεθος της λίστας ξεπεράσει έναν συγκεκριμένο αριθμό (1 εκατομμύριο παιδιά) η λύση δε βρίσκεται. Όταν βρεθεί τελική κατάσταση αντιγράφεται ο πίνακάς της στο schedule και μετά γράφονται τα δεδομένα σε txt και csv.

**4.2 Randomize**

Στη συγκεκριμένη συνάρτηση δεν λαμβάνονται υπ’ όψη οι ώρες διδασκαλίας των καθηγητών. Δεν μας ενδιαφέρει για αρχή να βάλουμε και σωστές συνολικά ώρες καθώς αυτό θα αλλαχθεί στη συνέχεια. Σκοπός μας είναι από τη μία να βάλουμε όλα τα μαθήματα με τις σωστές συνολικές τους ώρες σε τυχαίες θέσεις και να τους αναθέσουμε ένα τυχαίο καθηγητή. Παραβιάζονται δηλαδή πολύ περιορισμού οι οποίοι όμως θα λυθούν στη συνέχεια. Αρχικά χρησιμοποιούμε έναν πίνακα hours για να κρατάμε τις ώρες όλων των μαθημάτων. Με τη χρήση 3ων εμφωλευμένων επαναλήψεων μπαίνουμε σε κάθε θέση του πίνακα. Αρχικά το θέτουμε ως κενό (0,0). Μετά ελέγχουμε αν μας έχουν μείνει καθόλου ώρες από τα διαθέσιμα μαθήματα(κ[0,1,2] = Α, κ[3,4,5] = Β, κ[6,7,8] = Γ). Αν υπάρχει κάποιο εναπομένον μάθημα τότε επιλέγουμε κάποιο μάθημα από αυτά που απέμειναν τυχαία και εφόσον είναι του συγκεκριμένου τμήματος το βάζουμε στον πίνακα με set. Μετά ελέγχουμε αν ο κωδικός του συγκεκριμένου μαθήματος έχει ξανασυναντηθεί στον πίνακα. Αν ναι τότε αναθέτουμε τον ίδιο καθηγητή (5ος δυνατός περιορισμός). Αν όχι τότε ψάχνουμε για έναν τυχαίο καθηγητή που μπορεί να διδάξει το μάθημα. Δεν μας ενδιαφέρει αν ο καθηγητής που θα βάλουμε έχει εναπομείναντες ώρες. Αυτό το πρόβλημα θα λυθεί μετά κατά την εκτέλεση του Α\*. Προφανώς αν κανένας από τους προηγούμενους 3 ελέγχους δεν βγει αληθής τότε στη συγκεκριμένη θέση του πίνακα θα έχουμε κενό. Τέλος γίνεται επαναφορά των ωρών διδασκαλίας μαθημάτων από το hours επειδή κάθε φορά που αναθέτουμε ένα μάθημα στον πίνακα του μειώνουμε τις ώρες του από τον στατικό πίνακα lessons. Αφού ολοκληρωθεί η διαδικασία επιστρέφεται το schedule στην main.

**5.1 State**

Πρόκειται για τον κατασκευαστή των καταστάσεων οι οποίες θα περιέχουν έναν πίνακα 5x7x9 για το πρόγραμμα, το score το οποίο είναι η τιμή της ευρετικής για το συγκεκριμένο πρόγραμμα (κάθε δυνατός περιορισμός που παραβιάζεται δίνει στο σκορ +4000 και κάθε χαλαρός περιορισμός προσθέτει +1) το βάθος του State το οποίο έχει παραχθεί στο δέντρο, καθώς και την F. Για το συγκεκριμένο πρόγραμμα F=h(n) + g(n) όπου h(n)=score και g(n)=depth\*200. Άρα F=score+200\*depth. Η αρχικοποίηση της κατάστασης γίνεται με την εισαγωγή ενός πίνακα 5x7x9. Το σκορ, βάθος και F ανατίθενται μόνο με τη χρήση των set. Έτσι στην isTerminal απλά ελέγχουμε αν το σκορ είναι μικρότερο του 4000 το οποίο σημαίνει πως δεν παραβιάζεται κανείς δυνατός περιορισμός. Η F που περιέχει το άθροισμα όλων των παραμέτρων χρησιμοποιείται κατά την search για την επιλογή του καλύτερου παιδιού.

**5.2 Calculate Score**

Στην συνάρτηση CalculateScore υπολογίζουμε το σκορ με βάση τους χαλαρούς και δυνατούς περιορισμούς που παραβιάζονται και με τη βοήθεια των στατικών πινάκων lessons και teachers. Αρχικά ελέγχουμε για ύπαρξη κενών ανάμεσα σε μαθήματα. Προσμετρούνται και τα κενά σε αρχικές ώρες και γενικά κενά τα οποία συνοδεύονται από μάθημα. Αυτό δεν σημαίνει ότι, αν για παράδειγμα έχουμε 5 ώρες κενό και μετά τις 2 τελευταίες ώρες μάθημα, θα δοθεί +4000 στο σκορ, αλλά θα προσμετρηθούν όλα τα κενά συνολικά +20000 στο σκορ. Μετά βρίσκουμε περιπτώσεις όπου κάποιος καθηγητής διδάσκει για πάνω από 2 ώρες συνεχόμενα. Και πάλι προσμετρούνται όλοι οι παραβιασμένοι περιορισμοί. Για παράδειγμα αν διδάσκει 5 ώρες συνεχόμενα θα έχουμε αύξηση +12000 στο σκορ. Μετά βρίσκουμε αν δουλεύει κάποιος καθηγητής παράλληλα την ίδια ώρα σε διαφορετικά τμήματα και αυξάνουμε το σκορ ανάλογα με τον αριθμό παραβιάσεων. Στη συνέχεια για κάθε ώρα που κάποιος καθηγητής δουλεύει παραπάνω από το διαθέσιμο προσθέτουμε +4000 στο σκορ ολοκληρώνοντας έτσι τους δυνατούς περιορισμούς. Παρατηρείστε ότι δεν πήραμε υπ’ όψη τις συνολικές ώρες μαθημάτων και αυτό γιατί έχουμε φροντίσει από το randomize οι συνολικές ώρες να είναι ακριβώς ίσες με αυτές που δίνεται από την αρχή, αλλά υπολογίζουμε τις παραβιάσεις από τις ώρες καθηγητών τις οποίες δεν πήραμε υπ’ όψη στη συνάρτηση.

Συνεχίζουμε υπολογισμό του μέσου όρου διδασκαλίας όλων των καθηγητών που αποθηκεύεται στη μεταβλητή avg\_A (περιορισμός 2). Στη συνέχεια για κάθε καθηγητή ελέγχουμε αν ο αριθμός των διαθέσιμων ωρών διδασκαλίας του ξεπερνά το μέσο όρο και αν ναι τότε προσθέτουμε στο σκορ ανάλογα με το βαθμό παραβίασης. Βαθμός παραβίασης είναι η διαφορά των ωρών που διδάσκει ο καθηγητής με τον μέσο όρο των ωρών όλων των καθηγητών. Να σημειωθεί ότι αν για παράδειγμα ο μέσος όρος είναι 5 ώρες και ένας καθηγητής που έχει διαθέσιμες 10 ώρες διδάσκει μόνο 1 ώρα τότε θα προστεθεί ο αριθμός 4 στο σκορ. Γι’ αυτό χρησιμοποιούμε και την συνάρτηση Math.Abs. Μετά υπολογίζουμε για κάθε τμήμα κατά μέσο όρο πόσες ώρες διδασκαλίας πρέπει να έχει κάθε μέρα της εβδομάδας με βάση το συνολικό αριθμό ωρών διδασκαλίας των μαθημάτων της ανάλογης τάξης (Α,Β ή Γ) διαιρεμένο κατά 5 (περιορισμός 3). Για κάθε τμήμα έχουμε τις αντίστοιχες μεταβλητές avg\_A, avg\_B και avg\_C. Μετά, σε κάθε τμήμα ψάχνουμε και βρίσκουμε πόσες ώρες διδασκαλίας (χωρίς κενά) έχει κάθε μέρα και προσθέτουμε την διαφορά του μέσου όρου με τις ώρες διδασκαλίας της συγκεκριμένης ημέρας στο score. Τέλος, για κάθε μάθημα αναζητούμε το μέγιστο αριθμό ωρών διδασκαλίας και τον ελάχιστο σε κάθε τμήμα ανά ημέρα (περιορισμός 1). Αφού βρούμε το min και το max προσθέτουμε στο σκορ ανάλογα με το αποτέλεσμα της διαφοράς τους. Το σκεπτικό πίσω από αυτό είναι ότι εφόσον οι ώρες είναι καλά κατανεμημένες, ο ελάχιστος και ο μέγιστος αριθμός ωρών θα έχουν μικρή-μηδενική διαφορά.

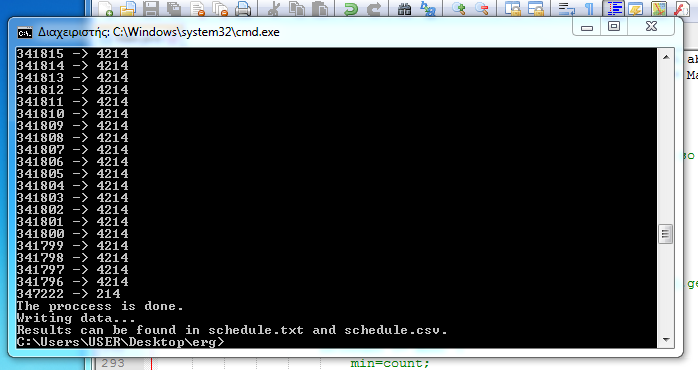
**5.3 Get Children**

Στη συνάρτηση getChildren δημιουργούμε μια λίστα που θα περιέχει όλα τα παιδία της τρέχουσας κατάστασης. Αρχικά για κάθε μάθημα πάνω στον πίνακα δημιουργούμε παιδία για κάθε πιθανή εναλλαγή καθηγητών. Για κάθε μάθημα, δηλαδή, βρίσκουμε τους καθηγητές που μπορούν να το διδάξουν και τους εναλλάσσουμε. Αυτό βέβαια σημαίνει ότι θα πρέπει για κάθε νέα επιλογή καθηγητή να αλλάξει ο διδάσκον από όλες τις ώρες στο συγκεκριμένο τμήμα και στο συγκεκριμένο μάθημα. Για παράδειγμα αν ο Νίκος δίδασκε 4 ώρες μαθηματικά και επιλέξουμε να τον αλλάξουμε τον Γιώργο τότε θα πρέπει ο Γιώργος να διδάσκει και στις 4 αυτές ώρες σύμφωνα με τον πρόσθετο περιορισμό που βάλαμε. Μέσα σε εμφωλευμένες επαναλήψεις ελέγχει αν το μάθημα ανήκει στο συγκεκριμένο τμήμα που εξετάζουμε και για κάθε καθηγητή που μπορεί να το διδάξει βρίσκει όλες τις θέσεις του πίνακα που είναι τοποθετημένο το μάθημα και κάνει την εναλλαγή. Η αλλαγή γίνεται με τη δημιουργία ενός νέου Compo με το ίδιο μάθημα αλλά νέο καθηγητή, το οποίο εισάγεται στο schedule. Μετά δημιουργείται η νέα κατάσταση, υπολογίζεται το σκορ, βάθος και F και προστίθεται στη λίστα.

Αφού τελειώσει η παραπάνω διαδικασία πρέπει να βρούμε και τα παιδιά που προκύπτουν από εναλλαγές μαθημάτων ίδιου τμήματος. Έτσι για κάθε θέση του πίνακα αλλάζουμε το Compo του με όλες τις επόμενες θέσεις. Φυσικά μόνο με επόμενες γιατί με τις προηγούμενες θέσεις έχει ήδη γίνει εναλλαγή από προηγούμενη επανάληψη. Ελέγχουμε φυσικά Αν πρόκειται για τα ίδια μαθήματα και αν ναι τότε δεν πραγματοποιείται αλλαγή μεταξύ τους. Κατά την αλλαγή δημιουργείται νέα κατάσταση και 2 νέα Compo στα οποία τοποθετούμε τα στοιχεία των 2 θέσεων του πίνακα και μετά τα βάζουμε στην αντίθετη θέση (swap). Αφού γίνει αυτό υπολογίζεται το σκορ, το βάθος και η F και τοποθετείται η νέα κατάσταση στη λίστα. Στο τέλος η λίστα με τα παιδιά επιστρέφεται.

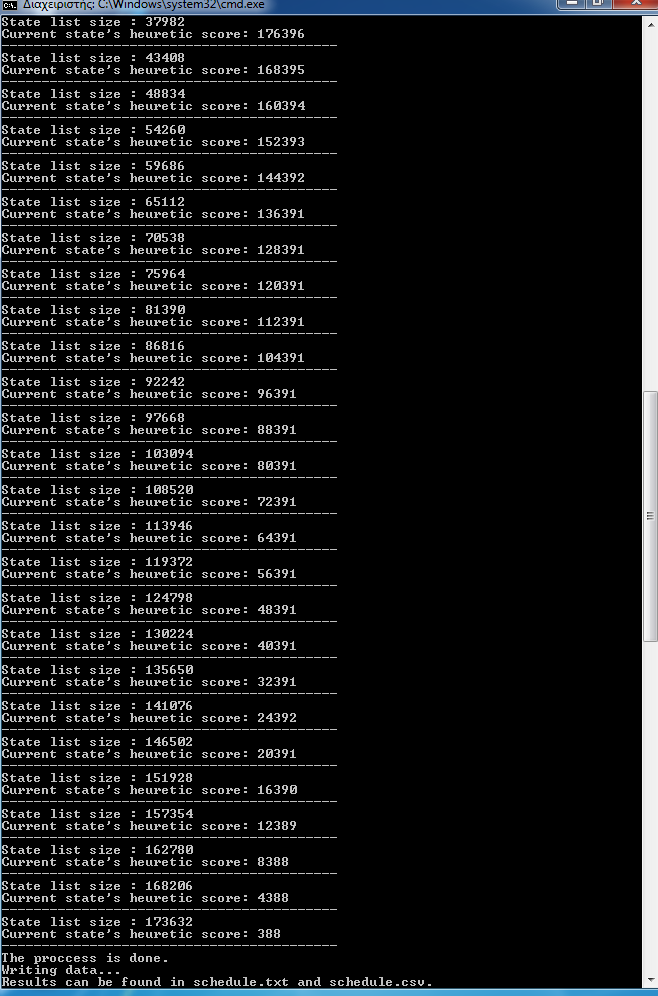
**6. A Star**

Περιέχει ένα array list για όλες τις καταστάσεις παιδιά και ένα hash set για να εντοπίζουμε τις καταστάσεις που έχουμε ήδη επισκεφτεί. Καλό θα ήταν να αναφερθεί ότι για να δουλέψει το hashing χρειάστηκε σε μερικές κλάσεις να κάνουμε Override τη συνάρτηση για Hash, ειδάλλως αν χρησιμοποιούσαμε την default συνάρτηση το hash code που θα είχε δημιουργηθεί θα είχε λάβει υπ’ όψιν του και το Score και το F,πράγμα που θα έκανε κάθε State να είχε διαφορετικό hash code το ένα από το άλλο. Αυτό που μας ενδιαφέρει εμάς να ελέγξουμε είναι μόνο το hash που παράγει το table. Η κύρια συνάρτηση εδώ είναι η search η οποία όπως είδαμε παραπάνω χρησιμοποιείται για τον εντοπισμό τελικής κατάστασης στην main. Αρχικά δέχεται μια αρχική κατάσταση η οποία τοποθετείται στη λίστα. Μέσα σε μια επανάληψη, όσο το μέγεθος της λίστας δεν είναι μηδενικό ή δεν έχει βρεθεί τελική κατάσταση, βρίσκουμε το ελάχιστο από τη λίστα (ανάλογα με το F), ελέγχουμε αν περιέχεται μέσα στο hash set και αν δεν περιέχεται τυπώνουμε τα στοιχεία του (μέγεθος λίστας, σκορ επιλεγμένης κατάστασης). Αν δεν είναι τελική κατάσταση παραγάγουμε τα παιδιά του και τα τοποθετούμε στη λίστα. Αν είναι τελική την επιστρέφουμε. Αν το μέγεθος της λίστας παιδιών είναι πολύ μεγάλο (έχουμε θέσει κάποιο όριο) τότε καθαρίζουμε την λίστα που περιέχει τα States καθώς και το hash set και επιστρέφουμε ένα State το οποίο έχει score=-100.Αυτό είναι αδύνατο να συμβεί αν δεν το κάνουμε χειροκίνητα και έτσι το χρησιμοποιούμε για να «πούμε» στο πρόγραμμα ότι η προσπάθεια απέτυχε. Όταν γίνει αυτό η search θα ξανά ξεκινήσει με καινούριο αρχικό State, όπως αναφέρθηκε στη main (μετά από συγκεκριμένο αριθμό παιδιών υπολογιστές με 4GB RAM δεν μπορούν να συνεχίσουν). Η συνάρτηση που χρησιμοποιείται για την εύρεση ελάχιστης κατάστασης είναι η FindMinScoreState η οποία βρίσκει και αφαιρεί την κατάσταση με το μικρότερο F λαμβάνοντας υπ’ όψιν όλες τις παραμέτρους (βάθος, σκορ).



**7. Πειραματικά Αποτελέσματα και Μέθοδοι Τεχνητής Νοημοσύνης**

Αρχικά πρέπει να σημειώσουμε κάποια πράγματα σχετικά με το teachers.txt. Σας έχουμε παραθέσει 2 αρχεία τέτοιου τύπου. Στο πρώτο (το ονομάσαμε teachers0.txt) ο συνολικός αριθμός ωρών διδασκαλίας των καθηγητών ισούται ακριβώς με τον συνολικό αριθμό ωρών διδασκαλίας που απαιτούν τα μαθήματα. Είναι ουσιαστικά η πιο δύσκολη επιλογή για εύρεση τελικής κατάστασης. Στο άλλο αρχείο (teachers.txt) το οποίο χρησιμοποιούμε τώρα έχουμε αυξήσει τον αριθμό των διαθέσιμων ωρών διδασκαλίας των καθηγητών. Έτσι είναι πιο εύκολη η εύρεση μιας καλύτερης min κατάστασης κατά το search και συνεπώς πιο γρήγορη η εκτέλεση του αλγόριθμου με λιγότερα συνολικά παιδιά. Χαρακτηριστικό παράδειγμα είναι η εικόνα που παραθέσαμε πιο πάνω. Όπως μπορείτε να δείτε στα τελευταία στάδια ο αλγόριθμος δυσκολεύεται να βρει τις τελικές αλλαγές που χρειάζονται. Αντιθέτως, με το νέο txt ο αλγόριθμος όταν βρεθεί σε τέτοιο στάδιο μπορεί να επιλέξει κάποιον άλλο καθηγητή που δεν έχει ολοκληρώσει τις ώρες του για να λύσει τον τελικό δυνατό περιορισμό. Και έτσι με την εκτέλεση βγαίνει ένα τέτοιο αποτέλεσμα:



(αλλάξαμε κάποια System.out.println() κατά την εγγραφή της αναφοράς)

Με μόλις 173632 παιδιά βρέθηκε η λύση σε αντίθεση με το προηγούμενο txt που απαίτησε 347222 παιδιά και 1-2 επαναλήψεις του randomize. Προφανώς οι 1-2 επαναλήψεις δεν σημαίνει ότι ο αλγόριθμος δεν μπορεί να βρει λύση αλλά ότι δεν μπορούμε να του αναθέσουμε περισσότερη RAM (έγινε χρήση 4GB) για να κατασκευάσει παραπάνω παιδιά. Προς το παρόν έχουμε θέσει το όριο σε 500.000 παιδιά ενώ για min requirement 8GB είχαμε θέσει 1.000.000.

**8. Μπορούν οι χαλαροί να υπερισχύσουν των δυνατών περιορισμών;**

Η απάντηση είναι όχι. Αυτό θα το αποδείξουμε εξετάζοντας έναν προς ένα τους χαλαρούς περιορισμούς:

* Στη χειρότερη περίπτωση θα έχω 5 μαθήματα ανά τμήμα που διδάσκονται 7 ώρες το κάθε ένα τοποθετημένα σε διαφορετικές ημέρες. Έτσι το σκορ που θα παραγόταν θα ήταν 7\*5\*9 και αυτό γιατί θα υπολογιστούν για κάθε μάθημα οι ελάχιστες και οι μέγιστες ώρες που διδάσκεται. Μετά η διαφορά τους θα προστεθεί στο σκορ. Οπότε από αυτό βρίσκουμε +315. (χαλαρός 1)
* Έστω ότι σε κάθε τμήμα έχουμε 1 μέρα η οποία είναι γεμάτη κενά και κάποια άλλη γεμάτη μαθήματα. Θα έχουμε min=0 και max=7 με αποτέλεσμα για κάθε τμήμα να προστίθεται στο σκορ το 7. Άρα συνολικά προστίθεται +63 ακόμα στο σκορ. (χαλαρός 3)
* Ο αριθμός των μέγιστων διδακτικών ωρών είναι 315. Έστω μια περίπτωση κακής ένας καθηγητής διδάσκει και τις 315 ώρες. Το σκορ θα γίνει ως εξής: score = score + (315 – 0). Σε όλους τους άλλους καθηγητές προστίθεται 0. Άρα το συνολικό ποσό είναι 315. Το κρατάμε αυτό και προχωράμε σε μια άλλη περίπτωση. Έστω ότι έχουμε 315 καθηγητές με 1 ώρα διδασκαλίας ο καθένας. Για κάθε καθηγητή θα γινόταν η εξής πράξη: score = score + (1 – 0). Άρα συνολικά θα είχαμε 315. Αυτά είναι 2 ακραία παραδείγματα. Έστω ότι έχουμε 16 καθηγητές που διδάσκουν ίσες ώρες και 1 ο οποίος δεν διδάσκει καθόλου. Ο μέσος όρος ωρών είναι 19. Ο κάθε καθηγητής διδάσκει 21 ώρες. Οπότε για τους καθηγητές που διδάσκουν θα είχαμε την πράξη: score = score + (21 – 19). Για τον καθηγητή που δεν διδάσκει θα είχαμε: score = score + |0 – 19|. Συνολικά στο σκορ θα προστεθεί το 49. Άρα μπορούμε να πούμε σχετικά ότι από αυτό τον περιορισμό θα προστεθεί ο αριθμός X στο σκορ όπου 0<= X <=315.

Από τις παραπάνω παρατηρήσεις βρίσκουμε ότι συνολικά οι χαλαροί περιορισμοί μπορούν να δώσουν +(≃700). Οπότε δεν διατρέχουμε κίνδυνο να βρεθούμε σε περίπτωση όπου το σύνολο των χαλαρών περιορισμών θα ισούται η θα είναι μεγαλύτερο από έναν δυνατό περιορισμό.

**9. Έξοδος Προγράμματος**

Μόλις η εκτέλεση του αλγορίθμου τελειώσει θα παραχθούν 2 αρχεία που θα περιέχουν τα στοιχεία του πίνακα της τελικής κατάστασης. Σας προτείνουμε να ελέγξετε το csv αρχείο το οποίο παρέχει μια καλύτερη όψη του τελικού προγράμματος με ονόματα αντί για κωδικούς κατανεμημένα σε στήλες. Συνήθως η συνολική εκτέλεση παίρνει κάπου στα 15-30 δευτερόλεπτα. Αντιθέτως η εκτέλεση του προηγούμενου teachers.txt παίρνει πολύ παραπάνω καθώς μπορεί να δοκιμάσει να επαναληφθεί 2-3 φορές και αυτό βγάζει νόημα καθώς είναι πιο περιορισμένο το πρόγραμμα ως προς τις κινήσεις του.

**10. Η ευρετική είναι αποδεκτή;**

Η ευρετική που υλοποιήσαμε, τουλάχιστον όσον αφορά τους βαριούς περιορισμούς, ψάχνει στον πίνακα και βρίσκει πόσοι βαριοί περιορισμοί παραβιάζονται χωρίς κάποια παραποίηση των αρχικών μας υποθέσεων. Σε κάθε αποτέλεσμα της ευρετικής θα πρέπει να γίνουν τουλάχιστον score%4000 κινήσεις για να φτάσουμε σε βέλτιστη κατάσταση.

Ας εξετάσουμε κάθε πιθανό ενδεχόμενο:

Α) Κενό ενώ δεν πρέπει να υπάρχει -> θέλει τουλάχιστον n βήματα όπου n = αριθμός κενών σε λάθος ώρα. Ο αλγόριθμος μας βγάζει ότι χρειάζονται n βήματα που είναι λιγότερα ή ίσα από τα βήματα που χρειάζονται.

Β) Καθηγητής διδάσκει πιο πολλές ώρες από όσο μπορεί -> θέλει τουλάχιστον n βήματα όπου n = αριθμός παραπάνω ωρών του καθηγητή. Ο αλγόριθμος μας βγάζει ότι χρειάζονται n βήματα που είναι λιγότερα ή ίσα από τα βήματα που χρειάζονται.

Γ) Καθηγητής διδάσκει πολλές συνεχόμενες ώρες (πάνω από 2) ->θέλει τουλάχιστον n βήματα όπου n = αριθμός παραπάνω ωρών διδασκαλίας. Ο αλγόριθμος μας βγάζει ότι χρειάζονται n βήματα που είναι λιγότερα ή ίσα από τα βήματα που χρειάζονται.

Δ) Μάθημα δεν διδάσκεται από έναν μόνο καθηγητή -> διδάσκεται από 2 ή περισσότερους, άρα στην καλύτερη θα πρέπει να αλλάξω τουλάχιστον n καθηγητές, όπου n = αριθμός έξτρα καθηγητών. Ο αλγόριθμος μας βγάζει ότι χρειάζονται n βήματα τα οποία είναι λιγότερα ή ίσα από τα βήματα που χρειάζονται.

Ε) Καθηγητής διδάσκει την ίδια στιγμή -> χρειάζονται τουλάχιστον τόσες αλλαγές όσες και οι ώρες που διδάσκει διπλά (λιγότερα ή ίσα βήματα με αυτά που όντως χρειάζονται).

**Υποσημείωση:** Όταν λέμε ότι χρειάζονται n βήματα, στο σκορ θα προστεθεί η τιμή n\*4000.

Όπως βλέπουμε σε κάθε κομμάτι που εξετάζει η ευρετική μας βγαίνει ότι ο αριθμός των βημάτων που πρέπει να γίνουν είναι πάντα μικρότερος ή ίσος από τα βήματα που χρειάζονται στην πραγματικότητα, άρα ο αλγόριθμος μας είναι μια υποεκτίμηση των πραγματικών βημάτων, πράγμα που τον κάνει αποδεκτή και συνεπής ευρετική.